Comput. & Ops. Res., Vol. 1, pp. 67—95. Pergamon Press, 1974. Printed in Great Britain.

A HYPERCUBE QUEUING MODEL FOR FACILITY LOCATION AND REDISTRICTING IN URBAN EMERGENCY SERVICES

RICHARD C. LARSON*
M.I.T., Cambridge, Mass., U.S.A.

Scope and purpose—This paper deals with certain spatially-oriented resource allocation problems experienced by police, fire, emergency medical, and other spatially distributed emergency service systems. Focusing on a system's *response units* (e.g., police cars, fire engines, ambulances), it develops a computer-implemented model for exploring the operational behavior of the system under various strategies affecting the locations of the response units and the methods by which they are assigned to callers requiring service. The model computes a mixture of performance measures that allows a system planner to focus simultaneously on several region-wide objectives while assuring that spatial inequities in the delivery of service are maintained at an acceptable minimum.

The overall objective is to devise a computationally useful tool that can assist the planner in improving system performance at a fixed manpower level. Since the cost of each response unit is considerable (usually exceeding $120,000 annually for one two-man around-the-clock patrol unit), and since the vast majority (over 90 per cent) of the cost is directly consumed by salaries and pensions, there is a great need for decision-aiding tools to help increase the productivity of these systems.

Abstract—This paper develops computationally efficient algorithms for studying the analytical behavior of a multi-server queuing system with distinguishable servers. The model is intended for analyzing problems of vehicle location and response district design in urban emergency services, includes interdistrict as well as intradistrict responses, and allows computation of several point-specific as well as area-specific performance measures.

1. INTRODUCTION AND SUMMARY

Only recently have the resource allocation problems of spatially distributed urban emergency service systems, as exemplified by police, fire and ambulance services, been subjected to serious analytical attention. Much of the recent work is reviewed by Chaiken and Larson[1]. A review of similar work in non-emergency urban services is provided by Revelle, Marks and Liebman[2].

Two important and related allocation problems of urban services are the "districting" problem and the "location" problem. Given a region with a certain spatial distribution of demands for service and given N response units that are spatially distributed throughout the region, the districting problem is often stated as follows: "How should the region be partitioned into areas of primary responsibility (districts) so as to best achieve some level or combination of levels of service?"

---

* Richard C. Larson is Associate Professor of Urban Studies and Electrical Engineering, Massachusetts Institute of Technology. He holds a B.S., M.S. and Ph.D. in electrical engineering and O.R. from MIT. Professor Larson's papers have appeared in *Proceedings of IEEE System Science and Cybernetics Group, Journal of Research on Crime and Delinquency, Journal of Urban Analysis, Management Science, Sloan Management Review,* and *Operations Research.* He is author of a book *Urban Police Patrol Analysis,* MIT Press, 1972, which was awarded the 1972 Lanchester Prize of ORSA. He has served as a member of the Science and Technology Task Force of the President's Commission on Law Enforcement and Administration of Criminal Justice (1966–67) and the Police Advisory Panel of the National Commission on Productivity (1973).

In the context of a spatially dispersed emergency ambulance service, a *district* for a particular ambulance would consist of a region in which calls for ambulance service would be handled by that ambulance, providing it is available when the call is received. If the district's ambulance is unavailable, then an out-of-district ambulance would be assigned by the ambulance dispatcher. If all N ambulances should be simultaneously busy, then the dispatcher enters the call in queue for later dispatch. In particular cities, the word district may be replaced by "response area", "response zone", or "dispatch zone". In some cases, more than one response unit may share the same district, thereby dividing the workload of the district; this occurs, for instance, if several ambulances are garaged at the same location.

In the case of police patrol, our use of the word district applies to a police car's "sector" or "beat", which is the area that the car patrols while not responding to calls for service. The police car's sector may or may not correspond to the region in which the car has primary responsibility for responding to calls for service; in other words, regions of primary patrol responsibility do not necessarily have to coincide with regions of primary call-for-service responsibility. Additionally, some cities provide "backup" cars to the regular sector cars, and these cars handle calls for service only when all the sector cars are simultaneously busy. Having no region of primary dispatch responsibility, the backup cars (which may be sergeant's cars or police wagons or other specialty units) often patrol a region covering several regular sectors.

The location problem, which is obviously closely related to the districting problem, is often stated as follows: "How should the N response units be located or positioned while not responding to calls for service?" In ambulance applications, it is usual to have one ambulance located in each of the N districts. Each location is fixed, corresponding to a garage, fire station house, point on a street, etc. In the case of shared districts, two or more ambulances may be stationed at the same location. For police patrol, the "location" of each unit is mobile, corresponding to the areas that the unit patrols in its sector. In order to specify statistically the unit's location, one must know the relative amounts of time that the patrol unit spends in the various parts of its sector.

Thus, in urban emergency services, the analysis of districting and location problems should include the possibility of overlapping (as well as disjoint) districts and mobile (as well as fixed) locations. Moreover, due to the dispatcher's desire to avoid delaying calls in queue, any analysis of these systems should include cross-district (or interdistrict) dispatches as well as intradistrict dispatches.

As applied to urban emergency services, most previous analyses of districting and/or location problems have suffered from at least three deficiencies. First, most have focused solely on *intra*district responses of units, while ignoring *inter*district responses and design issues that relate to interdistrict response. Second, most previous studies have focused on only one performance measure (usually mean region-wide travel time or a closely related measure), thereby ignoring many other performance measures that characterize the operational effectiveness of these systems. Third, most previous studies have failed to incorporate the probabilistic nature of an urban emergency service system which is due to the Poisson nature of the call arrival process and the variability in service times.

The purpose of this paper is to propose and develop a computationally useful model that overcomes each of these three limitations. The paper develops computationally efficient algorithms that allow one to evaluate numerically the performance characteristics of systems having up to 12 cooperating emergency response units. This requires the formula-

tion and solution of a set of simultaneous linear equations with up to 4096 unknowns.

The paper starts with a review of recent literature pertaining to urban facility location and redistricting, indicating the three recurring weaknesses in previous work that the current model attempts to overcome. Then, after providing a brief description of terminology and model assumptions, the focus is on an iterative efficient way of generating the state transition matrix for the associated continuous-time Markov process, given a geographical description of the region of interest and a dispatch criterion. The state space of the system, when nonsaturated, can be described as the vertices of the N-dimensional unit hypercube in the positive orthant; each vertex corresponds to a particular combination of units busy servicing calls.

It is found to be important to "tour" the hypercube vertices in a unit-step fashion, and an algorithm for generating an appropriate binary sequence is developed. Then the details of determining the optimal unit(s) to dispatch while touring the hypercube are provided. An analysis of the algorithm shows somewhat surprisingly that the time required of the touring algorithm, measured in execution time per step (or operations per hypercube vertex), actually decreases as the problem size increases. After having shown how to generate the matrix of coefficients of the equations of detailed balance, it is shown how to reduce the storage requirements for the coefficients by a factor of 200 or more for problems of practical size.

Next, an iterative solution procedure for obtaining the steady-state probabilities is developed that exploits many of the special properties of the hypercube model, including the fact that adding the probabilities of states lying along specified hyperplanes results in the well-known formulas for an $M/M/N$ queuing system. This fact is used to provide an initial set of probability estimates for the iterative solution procedure. By ordering the iterations in a particular way, a Gauss–Seidel method is derived that always keeps the sum of estimation errors along each such hyperplane equal to zero following each iteration. It is shown that the problem structure is such that one can guarantee convergence of the method with minimal round-off error.

The measures of performance computed analytically by the model include the following: *region-wide*: mean travel time, workload imbalance, and fractions of dispatches that are interdistrict dispatches; *response unit specific*: workload (measured in fraction of time busy servicing calls), mean travel time, fraction of responses of each response unit that are interdistrict; *district specific*: fraction of responses into each district that are interdistrict, mean travel time; *point specific*: mean travel time, fraction of calls handled by response unit $n, n = 1, 2, \ldots, N$. This mixture of performance measures allows one to focus simultaneously on several region-wide objectives while assuring that spatial inequities in the delivery of service are maintained at an acceptable minimum. The user has the option of assuming a zero-line capacity system, which effectively models the case in which "overflows" are treated by special reserve units (perhaps located outside the region of study), or an infinite-line capacity system, in which case delayed calls are handled in a first-come first-served manner by the regular units within the region.

Illustrative computational results using a PL/I program are presented for up to $N = 12$ response units, and the method appears very feasible for even such large problems.

A concluding section discusses promising extensions and generalizations.

For the convenience of the reader, Table 1 contains summary definitions of symbols frequently used in the technical developments.

Table 1. Summary of frequently used symbols

N	Total number of response units
J	Total number of geographical atoms
f_j	Fraction of region-wide workload generated from atom j, $j = 1, 2, \ldots, J$
τ_{ij}	Mean travel time from atom i to atom j, $i, j = 1, 2, \ldots, J$
l_{nj}	Probability that response unit n is located in atom j while available, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
λ	Average number of calls for service generated per hour from within the entire region of interest
μ^{-1}	Average service time per call
C_N	Vertices of N-dimensional unit hypercube in the positive orthant
B	A vertex contained in C_N
b_i	The i^{th} binary digit (from the right) in B, $i = 1, 2, \ldots, N$
$w(B)$	Weight of vertex B ($\sum b_i$)
$v(B)$	Numerical value of vertex B
H_n	n^{th} hyperplane from the origin (intersecting C_N), $n = 0, 1, \ldots, N$
d_{ij}	Hamming distance between two vertices B_i and B_j
λ_{ij}	Infinitesimal mean rate at which transitions are made from state i to state j, given the system is in state i
Λ	Matrix of infinitesimal transition rates (with $\lambda_{ii} \equiv -\sum_{j \neq i} \lambda_{ij}$)
η_{ij}	Total number of response units that are optimal, given state B_i and geographical atom j; $i = 0, 1, \ldots, 2^N - 1$; $j = 1, 2, \ldots, J$
q_{ij}	Number of the i^{th} closest unit to geographical atom j, given that all units are available; $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
t_{ij}	Mean travel time from district i to atom j, $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
$g_{i,N}$	Fraction of vertices for which the i^{th} best unit is the optimal unit, given a total of N units; $i = 1, 2, \ldots, N$
γ_N	Matrix density (measured in number of nonzero elements divided by the total number of elements)
ρ_n	Workload of unit n (measured in fraction of time busy servicing calls), $n = 1, 2, \ldots, N$
ρ_{nj}	Fraction of all dispatches that send unit n to atom j, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$

2. LITERATURE REVIEW

In order to place the work reported here in perspective, we briefly review the most relevant related work that has been reported within the past 12 years. One of the first efforts in the police area known to the author is the work of Smith[3], who used a computer gradient search technique to successively redesign sectors (patrol beats) from smaller cells (or atoms) so as to minimize mean *intra*sector travel time. No intersector interaction was assumed, and the method failed to converge for a 16-sector, 144-cell problem after 1 hr 55 min of computation time on an IBM 650 computer. The reason for a lack of convergence is suggested by the recent analytical models of Larson [Ref. 4, pp. 78–84, 106–113] which reveal surprising insensitivities of mean intrasector travel time to widely differing sector geometries.

In 1968 Gass[5] employed a heuristic technique first used for political redistricting by Hess *et al.*[6] to design police sectors (beats) in Cleveland. This technique focused on equalizing call rates from each sector and simultaneously keeping to a minimum the spatial irregularity of the sector, as measured by its moment of inertia. The method incorporated only a deterministic depiction of the system and neglected intersector cooperation.

Another study[7] focusing on the design of police patrol sectors also attempted to minimize mean intrasector travel time, but was based on an incorrect methodology. The authors first designed "patrol areas" (geographical atoms) of equal "crime potential" (which they

measured in call-for-service workload) and they claimed the following: "If the patrol areas (atoms) are then assigned to cars on a purely random basis (that is, if a car is equally likely to be in any one of a specified number of areas), probability theory predicts that patrol cars will be closer to the point of need when they are requested than under any other system of patrol assignment." (Parentheses added.) On the contrary, a fixed-point prepositioning, usually near but not necessarily coincident with the centroid ("center-of-mass") of a patrol region, is the patrol assignment that minimizes mean intrasector travel time. But most police administrators find such prepositioning unacceptable because of lost preventive patrol coverage. Thus, there is a need to develop a location and redistricting methodology that is compatible with *mobile* location services such as police departments as well as *fixed* location services such as fire departments and ambulance services.

A network approach was developed by Santone and Berlin[8] (1969) at the National Bureau of Standards for the evaluation of alternative fire station sites. Here the travel time characteristics of a city are determined by modeling travel time to occur along a network and employing shortest route algorithms. Again, all responses are assumed to be *intra*-district, and a fire house's district is assumed to be all points closer to that firehouse than to any other. The computed response time is weighted by the seriousness of a fire reported from each possible location. Their program, rather than including an algorithm to iterate automatically toward "optimality", is to be used iteratively by a planner to determine the (weighted) response time effects of alternative proposed sites and districts. Employing the program with a knowledge of political and social constraints, they arrived at recommended new locations and districts for the fire department of East Lansing, Michigan.

A continuation and elaboration of this approach by Schneider and Symons[9, 10] (1971) relies on man–machine interaction to locate ambulance dispatch centers and district a region. The alternative locations are nodes in a network and (again) travel is accomplished from point to point by following a minimal travel time path of the network. Again, a *district* about each located center is defined to be all points closer to that center than to any other; and, all responses are assumed to be *intra*district. No interdistrict interaction is considered. Here the objectives are to minimize mean travel time (always assuming intradistrict travel) while satisfying a constraint on maximum possible travel time. An encouraging part of this work is that human analysts, while interacting with a computer-driven cathode ray tube, seemed to be able, in ten or fewer iterations, to locate and design better districts than any heuristic algorithm that was tried.

A recent paper by Toregas *et al.*[11] (1971) viewed the location of emergency service facilities as a set covering problem, using a model similar to that first analyzed by Hakimi [12, 13]. Here the objective was to locate facilities on nodes of a network so as to minimize maximum possible travel time (or distance). Again, an emergency facility's *district* is the set of points closer to that facility than to any other and, reflecting a focus on intradistrict responses only, the authors state, "It is assumed here that each facility has response capability at all times." They derived a zero-one integer programming formulation to solve the stated problem.

Some recent work for small numbers of districts has overcome many of the objections associated with the above methods. Carter, Chaiken and Ignall[14] (1972) analyze the case of two fixed-position response units and rigorously derive the optimal districts for the two units, assuming a very general distance metric and a simplified form of interdistrict cooperation. The stochastic and interdistrict behaviors of the system are fully incorporated in their model. In addition, two measures of effectiveness are treated simultaneously: mean travel

time and workload imbalance, the latter measured as the difference in the fractions of time worked by the most-busy and least-busy units. Concurrent work by Larson and Stevenson [15] (1972) investigated several insensitivities of these and other location and districting models. But in all of this analytical work "optimal" districts have proven very difficult if not impossible to find for $N > 2$ units. Thus, although these models have provided useful insights into certain aspects of location and districting problems, they have not addressed computational problems that arise in practical situations with many response units.

A different approach was recently taken by Campbell[16] (1972) based on suggestions of Larson. Here, the multi-server queuing model employed in Refs. [14] and [15] that facilitates the study of probabilistic phenomena and interdistrict interaction is extended for up to $N = 6$ units and the steady-state equations are solved numerically on a computer. Then, in a spirit similar to that represented in Refs. [8–10], the user can examine the numerical results and relocate and/or redistrict accordingly. In this iterative fashion, a very reasonable set of locations and districts can be found, incorporating a rich mixture of system performance criteria. However, the storage and execution time requirements of Campbell's method grow enormously with N, making computations for $N > 6$ infeasible.

A main concern of this paper is the generation of new algorithms to make the computations for the model described by Campbell feasible for problems whose size is typically encountered in most cities. As a guideline, the average police precinct in New York City contains approximately nine radio-dispatched patrol cars. Thus, our goal is to devise a method that is feasible and practical for at least $N = 11$ or $N = 12$ units.

3. MODEL DESCRIPTION AND TERMINOLOGY

Central to the model is a geographical description of the region under study. This region is assumed to be partitioned into J *cells* or *geographical atoms*. The atoms can be as small as necessary to avoid unreasonable quantization error, and they can assume any geometric shape. Associated with atom j $(1 \leq j \leq J)$ is the fraction of region-wide workload f_j generated from within the atom. $\left(\sum_{j=1}^{J} f_j = 1.\right)$

Travel time statistics also play a major role in the model. All mean travel times are computed from a travel time matrix whose generic element is τ_{ij}, the mean travel time from atom i to atom j. In general, $\tau_{ij} \neq \tau_{ji}$. The numerical values of the τ_{ij}'s may reflect complications to travel such as one-way streets, barriers, traffic conditions, etc. If no matrix of τ_{ij}'s is obtainable for the period of time under study, then by specifying the centroid $(\bar{x}_i, \bar{y}_i)$ of each atom, one can approximate τ_{ij} to be $(|\bar{x}_i - \bar{x}_j| + |\bar{y}_i - \bar{y}_j|)/v$, where v is the effective response speed.*

The geographical depiction of the "location" of a response unit is general enough to model the fixed locations of fire units and ambulances and the mobile locations of police patrol units. This is accomplished by specifying a location matrix $L = (l_{nj})$, where l_{nj} is the *probability that response unit n is located in atom j while available or idle*† (or, equivalently, the fraction of available or idle time that response unit n spends in atom j). We require that L

* In the computer program one can selectively override this default for the case of intra-atom travel (involving τ_{ii}'s).

† Here "idle" is a convenient queuing theory term to reflect the activity between responding to calls for service. In actuality, the unit may be far from idle, perhaps performing crime preventive patrol in the case of police units or performing equipment maintenance or field inspections in the case of fire units.

be a stochastic matrix, i.e., for all n, $\sum_{j=1}^{J} l_{nj} = 1$. A fixed location unit would have $l_{nj} = 1$ for some (small)* atom j and $l_{nk} = 0$ for $k \neq j$. A mobile location unit would most likely have several l_{nj}'s nonzero. Note that within this structure it is very natural to allow mobile units to have overlapping districts or areas of responsibility; for instance, atom k would belong to overlapping districts if $l_{n_1 k} \neq 0$ and $l_{n_2 k} \neq 0$ for some n_1 and $n_2 \neq n_1$.

From a queuing point of view we assume that customers (*calls for service*) are generated from within the region in a Poisson manner at a mean rate λ per hr, with each atom j acting as an independent Poisson generator with mean rate λf_j. We assume there are N servers, which are the *response units* located within the region.

If one is not concerned with the identity of busy servers, the queuing system is simply the $M/M/N$ system. The user can specify whether the system has *zero line capacity* or *infinite line capacity*. The following assumptions are implied by the $M/M/N$ model: (1) exactly one response unit is assigned to every call that is serviced; (2) the service time of any response unit for any call for service has a negative exponential distribution with mean $1/\mu$;† (3) the service time is independent of the identity of the server, the location of the customer, and the history of the system; (4) for the zero-line capacity case, any call for service that arrives while all N response units are busy is either lost or (more likely in practice) serviced from outside the region or by special reserve units from within the region; (5) for the infinite-line capacity case, any call for service that arrives while all N response units are busy is entered at the end of a queue of calls which is depleted in a *first come, first served* (*FCFS*) manner.‡

To describe the more complicated state space in which the *identities* of busy (unavailable) and idle (available) response units are retained we need certain definitions and operations pertaining to binary numbers. A generic one-digit binary number is denoted as b. A generic ordered set of N one-digit binary numbers is given by $B \equiv \{b_N, b_{N-1}, \ldots, b_1\}$. The weight of B, denoted $w(B)$, is equal to $\sum b_i$, the number of binary "ones" in the set B. For each set B there corresponds a unique numerical value $v(B)$, given in decimal by

$$v(B) = b_N \cdot 2^{N-1} + b_{N-1} \cdot 2^{N-2} + \cdots + b_1.$$

Conversely, for each positive integer k there corresponds a unique set $B(k)$ such that $v[B(k)] = k$. Where the meaning is clear we shall occasionally perform "arithmetic" operations on the set B, recognizing that $v(B)$ is implied.

To each set $B = \{b_N, b_{N-1}, \ldots, b_1\}$ there corresponds a unique point or *vertex* in R^N with the i^{th} coordinate equal to b_i $(i = 1, \ldots, N)$. The set C_N of all 2^N such vertices is the set of vertices of the *N-dimensional unit hypercube* in the positive orthant. The n^{th} *integer hyperplane from the origin* (or simply the n^{th} *hyperplane*) is the set

$$H_n = \{B \in C_N : w(B) = n\}, \quad n = 0, 1, \ldots, N.$$

Binary set operations are defined as follows:

Or: $B_1 \cup B_2 = \{1\}$ *iff* $B_1 = \{1\}$ or $B_2 = \{1\}$.

And: $B_1 \cap B_2 = \{0\}$ *iff* $B_1 = \{0\}$ or $B_2 = \{0\}$.

* If one wishes to be exact about the fixed location, the atom j could be defined to be a point atom, i.e., one of zero area.

† Thus variations in service time that are due to variations in travel times are ignored.

‡ The results of the model are also applicable under several other queuing disciplines such as *last-come, first-served* and *random*.

Complement:
$$B' = \begin{cases} \{0\} \text{ if } B = \{1\}, \\ \{1\} \text{ if } B = \{0\}. \end{cases}$$

These definitions carry over in the obvious way to sets having more than one digit.

The *Hamming distance* between two vertices B_i and B_j is the weight of the symmetric set difference

$$d_{ij} \equiv w([B_i \cap B_j'] \cup [B_i' \cap B_j]).$$

Thus d_{ij} is equal to the number of elements (binary digits) of the set B_i which differ from the corresponding elements in the set B_j. Geometrically, the Hamming distance is simply the rectilinear or "right-angle" distance between two points in R^N. We also find it convenient to define the "upward" Hamming distance

$$d_{ij}^{+} \equiv w(B_i' \cap B_j)$$

and the "downward" Hamming distance

$$d_{ij}^{-} \equiv w(B_i \cap B_j').$$

Clearly,

$$d_{ij} = d_{ij}^{+} + d_{ij}^{-}.$$

4. THE STATE TRANSITION MATRIX AND EQUATIONS OF DETAILED BALANCE

The state space of the zero-line capacity queuing model is the N-dimensional hypercube C_N, where each vertex (state) corresponds to a particular combination of response units busy and idle. Given a state $B = \{b_N, b_{N-1}, \ldots, b_1\}$, unit n is said to be *busy* if $b_n = 1$ and *idle* if $b_n = 0$. This state space is augmented by an "infinite tail" if one wishes to model the infinite-line capacity case. Since the two models are governed by the same equations for unsaturated states (i.e., states with at least one available response unit), we will focus the development on just one of the models: the zero-line capacity model. In Section 13 we indicate the necessary model modifications to allow for infinite line capacity.

A central problem is converting the geographical data into a queuing framework. To do this we define the state transition matrix $\Lambda = (\lambda_{ij})$, where λ_{ij} = infinitesimal mean rate at which transitions are made from state i to state j, given that the system is in state i; $i, j = 0, 1, \ldots, 2^N - 1$, $i \neq j$, and $\lambda_{ii} = -\sum_{j \neq i} \lambda_{ij}$. Here for convenience we index the states (vertices) according to their numerical values, i.e., we select i so that $v(B_i) = i, i = 0, 1, \ldots, 2^N - 1$. Note that Λ is a differential matrix (i.e., $\sum_j \lambda_{ij} = 0$).

There are two classes of transitions on the hypercube: *upward* transitions that change a unit's status from available to unavailable and *downward* transitions that do the reverse. For a given vertex $B_i = \{b_N, b_{N-1}, \ldots, b_1\}$, upward transitions can occur to all "adjacent" vertices B_j for which $d_{ij}^{+} = 1$. If unit n is the unit whose status is changed (from idle to busy), then $B_i = \{b_N, b_{N-1}, \ldots, 0_n, \ldots, b_1\}$ and $B_j = \{b_N, b_{N-1}, \ldots, 1_n, \ldots, b_1\}$. Downward transitions can occur to all adjacent vertices B_j for which $d_{ij}^{-} = 1$. A key observation is that no transitions occur to vertices that are more than unit Hamming distance from B_i. This is due to the fact that only one unit is assigned to each call.

Since the service times are all distributed as negative exponential random variables with mean μ^{-1}, the transition rate associated with each downward transition is equal to μ. Thus, for all (i,j) for which $d_{ij}^{-} = 1$ we have $\lambda_{ij} = \mu$. For convenience we set $\mu = 1$, thereby equating the unit of time to the mean service time.

The upward transition rates depend in a complicated way on the region's geography, the system state and the dispatching selection criterion. We will develop a recursive method to generate the set of upward transition rates, first by fixing the geographical atom of the call, then by *touring* the hypercube in a *unit-step* fashion. The entire matrix is completed as soon as the hypercube has been toured once for every geographical atom.

Since the model assumes Poisson input and negative exponential service times, thereby making knowledge of past system history irrelevant, the state of the system is fully specified by B_i. The model is thus a *finite-state continuous time Markov process** whose steady-state probabilities are determined from the *equations of detailed balance*,

$$P\{B_j\}[\lambda_j + w(B_j)] = \sum_{\{B_i \in C_N : d_{ij}^{+} = 1\}} P\{B_i\}\lambda_{ij} + \sum_{\{B_i \in C_N : d_{ij}^{-} = 1\}} P\{B_i\}, \quad j = 0, 1, \ldots, 2^N - 1 \tag{1}$$

where $P\{B_j\} \equiv$ Prob {system is occupying state j under steady-state conditions},

$$B_j \in C_N, j = 0, 1, \ldots, 2^N - 1; \text{ and}$$

$$\lambda_j = \begin{cases} 0 \text{ for } j = 2^N - 1, \\ \lambda \text{ otherwise.} \end{cases}$$

Heuristically, these equations require that the steady-state rate of transitions out of state B_j [the LHS of (1)] be equal to the steady-state rate of transitions into state B_j [the RHS of (1)], where the latter transitions include $w(B_j)$ upward transitions to B_j that result in a new unit becoming busy and $[N - w(B_j)]$ downward transitions to B_j that result in a new unit becoming idle. To guarantee a probability distribution, thereby eliminating a degenerate solution, we also require that the probabilities sum to one, i.e.,

$$\sum_{i=0}^{2^N - 1} P\{B_i\} = 1, \tag{2}$$

a condition which makes any one of the balance equations redundant and therefore removable from the set of equations.

Theoretically, the solution to this set of equations requires only a matrix inversion, and thus is relatively straightforward. However, the size of the matrix Λ (measured in total number of elements) is equal to 2^{2N}, thus becoming huge for even moderate values of N. For instance for $N = 10$ the matrix contains 1,048,576 elements. Thus one is confronted with both a computational and a storage problem of large magnitude.

5. GENERATING THE UPWARD TRANSITION RATES

We develop in this and the next section an efficient method for generating the upward transition rates. For each geographical atom j $(1 \leq j \leq J)$ we shall tour the hypercube in a unit-step fashion. From one step to the next, the status of only one response unit changes,

* See, for example, Ref. [17].

either from *busy* to *idle* or from *idle* to *busy*. The method for constructing such a tour is given in the next section.

Now, ignoring "ties" for the moment, it is quite likely that the optimal* unit to dispatch in the current state on the tour is either the same unit which was optimal in the state considered just prior to the current one, or the unit whose status has just changed, provided that unit has switched from busy to idle.

At each vertex we must compute the identities of the optimal units to dispatch. In particular, the following variables must be calculated:†

$\eta_{ij} \equiv$ total number of response units that are optimal, given state B_i and geographical atom j.

$r_l \equiv$ the number of the l^{th} optimal response unit, where $r_l = 1, 2, \ldots, N$ and $l = 1, 2, \ldots, \eta_{ij}$.

If $\eta_{ij} > 1$, it is assumed that the unit dispatched to atom j when the state is B_i is to be chosen *randomly* from among the units $r_1, r_2, \ldots, r_{\eta_{ij}}$.

Assuming now that the tour is constructed and the optimal units to dispatch are known, the upward λ_{ik} are calculated as follows. The matrix Λ is initialized to zero. Then, at the point on the tour associated with atom j and vertex $B_i = \{b_N, b_{N-1}, \ldots, b_1\}$, exactly η_{ij} transition rates will be incremented by $\lambda f_j/\eta_{ij}$. In particular,‡

$$\lambda_{ik} \leftarrow \lambda_{ik} + \lambda f_j/\eta_{ij} \tag{3}$$

for all adjacent states $B_k = \{b_N, b_{N-1}, \ldots, b'_{r_l}, \ldots, b_1\}$ such that $d_{ik}^{+} = 1$ and $b'_{r_l} = 1 \neq b_{r_l}$. Very frequently there are no ties (i.e., $\eta_{ij} = 1$), in which case only one addition is performed for every vertex for each geographical atom.

6. GENERATING A UNIT-HAMMING DISTANCE BINARY SEQUENCE

In order to tour the hypercube in a unit-step manner, one is confronted with the problem of generating a complete sequence $S_1, S_2, \ldots$ of N-digit binary numbers, with 2^N unique members in the sequence and with adjacent members being exactly unit-Hamming distance apart. Such a sequence represents a complete unit-step tour of the hypercube.

Identifying the binary numbers with their values, the following algorithm generates such a sequence $S_1, S_2, \ldots, S_{2^N}$:

Step 1: Set $S_1 = 0$, $S_2 = 1$, $m_2 = 2$, $n = 2$.

Step 2: If $n > N$, *STOP*, Binary sequence completed.

Step 3: $m_1 = m_2$, $m_2 = 2 \cdot m_1$, $i = m_1$.

Step 4: $S_{i+1} = m_1 + S_{m_2 - i}$.

Step 5: $i \leftarrow i + 1$.

Step 6: If $i < m_2$, go to step 4.

Step 7: $n \leftarrow n + 1$.

Step 8: Go to step 2.

---

* Various optimality criteria are discussed in Section 7.

† In Section 7, this calculation is illustrated for one set of assumptions.

‡ The operation "$\alpha \leftarrow \beta$" implies that the contents of the register whose value is "α" is replaced by a value "β".

Proof: (Induction) Clearly S_1 and S_2 represent a unit-Hamming distance complete sequence of 1-digit binary numbers. Now one shows that if a unit-Hamming distance complete sequence of k-digit binary numbers has been generated ($k = 1, 2, \ldots$), then steps 3 through 7 with $n = k + 1$ generate additional (new) members in the sequence to complete a $(k + 1)$-digit sequence. At step 3 one has $m_1 = i = 2^k$, $m_2 = 2^{k+1}$. Thus, the l^{th} new member is

$$S_{(2^k+l)} = 2^k + S_{(2^{k+1}-2^k-l+1)}, \quad l = 1, 2, \ldots, 2^k.$$

Letting d_{ij} be the Hamming distance between S_i and S_j, we clearly have $d_{(2^k)(2^k+1)} = 1$. Since by induction the sequence of k-digit binary numbers is a unit-Hamming distance sequence, we must have $d_{(2^k+l)(2^k+l+1)} = 1$, where $l = 1, 2, \ldots, 2^k - 1$, so the new members are unit-Hamming distance apart. Since by induction all 2^k members in the k-digit sequence are unique, then all 2^k new members are unique and are clearly greater than or equal to 2^k and strictly less than 2^{k+1}. Thus the new members complete a $(k + 1)$-digit sequence.

As an example, the following sequence is generated for $N = 3$:

Step 1:
Initialization

$S_1 = \{0, 0, 0\}$
$S_2 = \{0, 0, 1\}$

Steps 3–6:
Add "one" in 2nd position and step backwards

$S_3 = \{0, 1, 1\}$
$S_4 = \{0, 1, 0\}$

Steps 3–6 again:
Add "one" in 3rd position and step backwards

$S_5 = \{1, 1, 0\}$
$S_6 = \{1, 1, 1\}$
$S_7 = \{1, 0, 1\}$
$S_8 = \{1, 0, 0\}$

Stop.

This tour is depicted on a three-dimensional hypercube in Fig. 1.

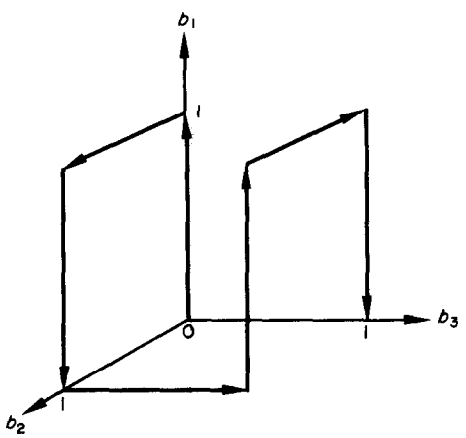

Fig. 1. Unit-step tour of the three-dimensional hypercube ($b_i = i^{\text{th}}$ binary digit = 0, 1).

It is worth noting that the generated tour is only one of many possible unit-Hamming distance complete tours of the hypercube. Moreover, any complete unit-Hamming distance k-digit sequence can be provided to the algorithm at step 3 with $j = k + 1$ and $m_2 = 2^k$, and the algorithm will generate complete sequences whose members have any larger number of digits $(k + 1, k + 2, \ldots)$.

7. DETERMINING THE OPTIMAL UNIT(S)

Now that we can tour the hypercube in a unit-step fashion, we must determine at each vertex the identity of the best available unit(s) to dispatch. For a certain class of dispatch policies, it turns out that this can be done very efficiently by exploiting the "unit-step" property. This class includes any policy having the property that the dispatch preferences for units to calls from any particular atom can be determined without reference to system state. In other words, for such a policy one can always say that some unit n_1, if available, would be the first preference to dispatch to atom j, unit n_2 would be the second preference, unit n_3 the third preference, etc. For such "fixed preferences" policies the dispatcher always selects the most preferred *available* unit, given the system state. Thus, for instance, the dispatch selection criterion may be strict center-of-mass (SCM), modified center-of-mass (MCM), expected strict center-of-mass ($\overline{\text{SCM}}$), or expected modified center-of-mass ($\overline{\text{MCM}}$) [4], and it may include additional complications, such as giving the district's "own" unit first preference regardless of estimated travel times.

More complicated dispatch policies such as time-average minimization of system-wide travel time (as illustrated by Carter, Chaiken, and Ignall[14]) are not included in the class of policies we are considering. For such policies the determination of the optimal unit may be a very difficult task, involving many more calculations than illustrated by the techniques of this and the next section. However, the results of such calculations can be fed directly into the hypercube model, thereby facilitating the computation of steady-state probabilities and system performance measures.

In order to be concrete (and brief) in the following discussion, we develop the dispatch selection algorithm in the context of one particular "fixed preference" policy, namely expected modified center-of-mass ($\overline{\text{MCM}}$). With this strategy the dispatcher takes into account the geographical atom of the call and correctly utilizes all the information he has available regarding locations of units. Defining

$$q_{ij} \equiv \text{the number of the } i^{\text{th}} \text{ closest unit to } \textit{geographical atom } j, \text{ given that all units are available,}$$

$$t_{ij} \equiv \text{mean travel time from district } i \text{ to } \textit{geographical atom } j = \sum_k l_{ik}\tau_{kj},$$

the dispatcher using an $\overline{\text{MCM}}$ policy will select that available unit with minimum t_{ij}. In the case of a tie, the tie is broken by random choice.

Suppose we are in the process of touring the hypercube and consider a step from vertex S_{i-1} to vertex S_i. We have already determined the identities of the $\eta_{(i-1)j}$ units that are optimal units at vertex S_{i-1}, assuming a call from geographical atom j. Now, the step can cause any one of the N units to change status. The particular unit whose status is changed is

$$n_0 = \log_2(v[\{S_{i-1} \cap S_i'\} \cup \{S_{i-1}' \cap S_i\}]) + 1.$$

If $v(S_{i-1}) > v(S_i)$ then unit n_0 is now available; otherwise it is now unavailable.

Consider the former case first. There are three possibilities:

(1) Unit n_0 is now the unique optimal unit;

(2) Unit n_0 is an additional optimal unit;

(3) Unit n_0 is not an optimal unit.

Given the $\overline{\text{MCM}}$ dispatching criterion, the possibility that applies is readily determined by comparing the travel time of the new unit $t_{n_0 j}$ to $t_{r_1 j}$, the travel time of a previously optimal unit. If either (1) or (2) applies, then certain status variables (e.g., η_{ij}, $r_{\eta_{ij}}$) must be changed. If (3) applies, then no further computations are required. This completes operations at vertex S_i if unit n_0 is now available.

Now consider the case in which unit n_0 is unavailable. There are also three possibilities here:

(1) Unit n_0 was the unique optimal unit.

(2) Unit n_0 was one of the two or more optimal units.

(3) Unit n_0 was not an optimal unit.

Again, the situation which applies is determined by comparing $t_{n_0 j}$ to $t_{r_1 j}$. If (3) applies, nothing further happens. If (2) applies, then unit n_0 is removed from the list $r_1, r_2, \ldots, r_{\eta_{ij}}$, and the value of η_{ij} is decreased by one. If (1) applies, then another optimal unit (or set of optimal units) must be found. This is done by searching the list q_{ij}, starting immediately after the entry for which $q_{ij} = n_0$. The first available unit that is found from this list is an optimal unit for vertex S_i. This unit now becomes r_1. Additional units in the list of q_{ij} must be examined since the possibility of a tie exists. This additional search is terminated as soon as a t_{ij} is found that is strictly greater than $t_{r_1 j}$. At this point all the r_l's and η_{ij} have been computed, and the job is finished.

There exists one further complication for situation (1) above, and that involves the case for which vertex B_i is $\{1, 1, 1, \ldots, 1, 1\}$. Occurring once each tour, this is the state in which all N units are simultaneously unavailable. Obviously, any search for available units will be fruitless. But we want to preserve the generality of the algorithm and avoid a situation in which we would have to test continually for state $\{1, 1, 1, \ldots, 1, 1\}$ and invoke special procedures once it is incurred. The problem is solved by defining an "artificial" unit, designated unit "$N + 1$", which is always available for an N-unit problem. We make unit $N + 1$ particularly unattractive to dispatch by setting $t_{(N+1)j} = +\infty$ for all j.* Thus, if the algorithm finds $B_i = \{1, 1, \ldots, 1\}$, it reluctantly selects unit $N + 1$ as the optimal unit. One step later, when it finds that unit $N + 1$ was the optimal unit and that some other unit n ($n = 1, \ldots, N$) is available, it immediately designates unit n as the optimal unit for the new vertex.

The PL/I listing, called "TOUR", which implements the above algorithm, is given in a forthcoming document.

Regarding the generality of the TOUR algorithm, one notes that the dispatch criterion enters only at the point of determining which of the three possibilities applies, given either the availability or unavailability of the unit n_0. Thus the general structure of the algorithm remains invariant within the large class of "fixed preferences" dispatch policies. To adapt the algorithm to another dispatch policy within this class, one only need replace the comparison of $t_{n_0 j}$ to $t_{r_1 j}$ with the analogous comparison or other procedure associated with the new criterion.

* In actuality, "$+\infty$" in a computer program corresponds to a very large yet finite number.

8. ANALYSIS OF THE TOUR ALGORITHM

In this section we briefly analyze the efficiency of the TOUR algorithm. We find that each execution of the algorithm actually becomes less time consuming as reflected by the number of operations required at each vertex, as the problem size (reflected by N) increases!

For simplicity in the analysis we ignore the possibility of ties, an assumption that is quite reasonable in most applications. We note that the "work" required of TOUR is invariant to the particular geography at hand and the values that the state probabilities will assume. The basic tree structure that depicts the execution of the algorithm is given in Fig. 2. There are four possibilities, corresponding to whether the unit whose status has just changed is now available or unavailable and optimal or not optimal. Above each branch in the tree is the conditional probability that that branch will apply. For instance, 50 per cent of all unit-steps result in a new unit becoming available; thus we have "1/2" above the two left-most branches, indicating "available" and "unavailable". Supposing a new unit is available, the probability that it is optimal is simply $1/N$. Implicit in this computation is the assumption that the unit-step sequence is generated independently of the problem's particular geography, an assumption that appears very reasonable. Similar conditional probabilities apply for the case in which a unit becomes unavailable. Below each branch in Fig. 2 is the number of operations (computations) required to perform that branch. For instance, one comparison is required to determine whether the new unit is available or unavailable; similarly, one comparison is needed with $\overline{\text{MCM}}$ (or any other "fixed preferences" dispatch policy) to determine whether the new unit is optimal for dispatch. When a change in the optimal unit is found, a fixed number n_{UD} of update operations is performed; this is typically two or three simple substitutions.

The one final issue we must investigate in order to complete analysis of the TOUR algorithm is the statistical behavior of the variable n_S, the number of operations required in a search for a new optimal unit. (See Fig. 2.) This search occurs in the ordered list q_{ik} and starts immediately after that value of i corresponding to the unit that was optimal and is no longer available. Assuming one operation for each unit whose status is checked, the value

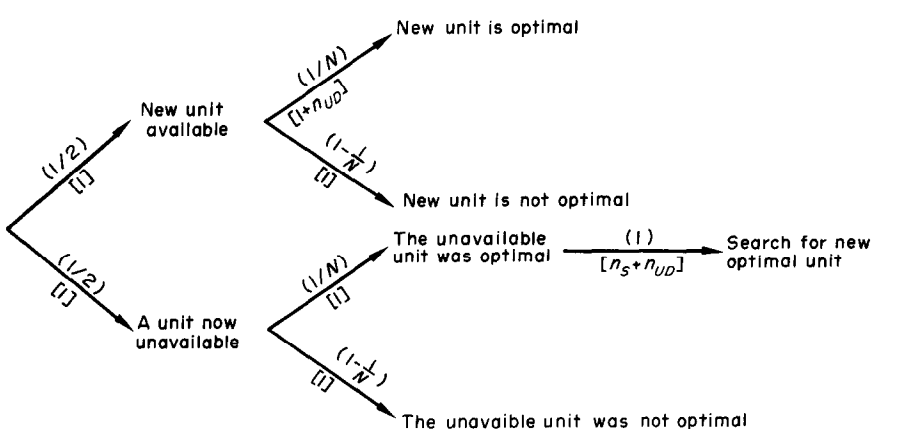

Fig. 2. Sequential tree depicting execution of TOUR algorithm.

of n_S is equal to one plus the number of successive unavailable units incurred in the list q_{ik}.* To determine the statistics of n_S it is convenient to define

$$g_{i,N} \equiv \text{fraction of vertices for which the } i^{\text{th}} \text{ best unit is the optimal unit, given a total of } N \text{ units, } i = 1, 2, \ldots, N, N+1.$$

Clearly,

$$g_{1,N} = \frac{1}{2},\ g_{2,N} = \frac{1}{4}, \ldots, g_{i,N} = \frac{1}{2^i}, \ldots, g_{N,N} = \frac{1}{2^N},\ g_{N+1,N} = \frac{1}{2^N}.$$

Here $g_{N+1,N} = 1/2^N$, corresponding to the vertex $\{1, 1, \ldots, 1\}$, in which case the artificial $N+1^{\text{st}}$ unit is selected as optimal. Now n_S is closely related to the "mean depth" of the algorithm, as given by

$$\bar{e}_N \equiv \sum_{i=1}^{N+1} i g_{i,N}.$$

Performing the computation we find

$$\bar{e}_N = 2 - \left(\frac{1}{2}\right)^N,$$

implying that the mean depth of the algorithm converges exponentially to 2. Loosely speaking, for large N, this says that the "average assignment"† is to the second preferred unit. Referring now to n_S, if the i^{th} preferred unit was optimal and is no longer available, there are $N - i$ actual units and one artificial unit whose status the algorithm may have to interrogate in a search for the next optimal unit. But the mean depth of this conditional search is simply $\bar{e}_{N-i}$. Now the probability that the i^{th} preferred unit was optimal and is no longer available is proportional to $g_{i,N}$, the fraction of vertices for which the i^{th} best unit is the optimal unit, for all $i = 1, 2, \ldots, N$. We must exclude unit $N+1$ from consideration since it never becomes unavailable. Thus the average value of the variable of interest is

$$\begin{aligned}
\bar{n}_S &= \sum_{i=1}^{N} \left[\frac{g_{i,N}}{1 - g_{N+1,N}}\right] \bar{e}_{N-i} \\
&= \sum_{i=1}^{N} \left[\frac{\frac{1}{2^i}}{1 - \left(\frac{1}{2}\right)^N}\right] \left[2 - \left(\frac{1}{2}\right)^{N-i}\right] \\
&= 2 - \frac{N}{2^N - 1}. \qquad (4)
\end{aligned}$$

Thus, the unconditional average value of the number of operations required in a search for a new optimal unit converges very quickly (from below) to 2.

* In defining n_S, the artificial unit $N+1$ can be treated in a number of ways. The current definition treats unit $N+1$ just like any actual unit, assuming the algorithm must check its status even though we know that unit $N+1$ is always available for an N-unit problem.

† Here the "average" is taken over the collection of hypercube vertices and has nothing to do with the state probabilities associated with those vertices.

We are now in a position to compute the unconditional average total number of operations required to execute the TOUR algorithm once (i.e., at one vertex). Employing the above results and the conditional probabilities given on the tree in Fig. 2, we obtain for this average value

$$\bar{n}_{\text{TOUR}} = \frac{1}{2}\left[1 + \left(1 - \frac{1}{N}\right)\cdot 1 + \frac{1}{N}(1 + n_{UD})\right] + \frac{1}{2}\left[1 + \left(1 - \frac{1}{N}\right)\cdot 1 + \frac{1}{N}\left(1 + \left\{2 - \frac{N}{2^N - 1}\right\} + n_{UD}\right)\right]$$

$$= 2 + (n_{UD} + 1)/N - \frac{1/2}{2^N - 1}. \tag{5}$$

What is surprising about this result is the fact that TOUR requires least time to execute for very large problems!

As an example, for a three-unit problem with $n_{UD} = 3$, $\bar{n}_{\text{TOUR}} = 137/42 \approx 3{\cdot}26$ operations. For a ten-unit problem again with $\bar{n}_{UD} = 3$, $\bar{n}_{\text{TOUR}} \simeq 2{\cdot}4$ operations, a reduction in effort per execution of approximately 26 per cent. Of course, the total number of vertices grows as 2^N, so the reduction per vertex does not dramatically alter the exponential growth of the entire problem size with N.

9. STORING THE STATE TRANSITION MATRIX

The TOUR algorithm speeds execution time and eliminates the need to store very large intermediate matrices. There is one additional storage problem to be confronted, that of storing the state transition matrix Λ, containing 2^{2N} elements. This problem is readily solved once one exploits certain sparcity and regularity properties of the matrix. Consider column j of the matrix: it contains exactly $w(B_j)$ terms corresponding to upward transition rates (i.e., λ_{ij}'s such that $d_{ij}^{+} = 1$); $[N - w(B_j)]$ terms that are equal to unity, the service rate; and an on-diagonal term λ_{jj}. All other terms, corresponding to transition rates from states greater than unit-Hamming distance away from B_j, have value zero. Thus, each column contains only $N + 1$ nonzero entries, yielding a *matrix density* (measured in number of nonzero elements divided by the total number of elements) of

$$\gamma_N = \frac{(N + 1)2^N}{2^{2N}} = \frac{N + 1}{2^N}. \tag{6}$$

For $N = 10$, for instance, the density is $\gamma_{10} \approx 0{\cdot}0107$, reflecting an exceedingly sparce matrix.

Since all the service rates are unity, one only has to store the upward transition rates and the on-diagonal terms, resulting in a further efficiency of nearly 50 per cent. The storage procedure is as follows:

(1) Designate MAP(j) as the address in array A of the on-diagonal term λ_{jj} corresponding to state B_j, i.e., A[MAP(j)] $= -[\lambda_j + w(B_j)]$, where $\lambda_j = \lambda$ for all $j = 0, 1, 2, \ldots, 2^N - 2$ and $\lambda_j = 0$ for $j = 2^N - 1$.

(2) By generating the set of adjacent states successively by changing the status of unit 1, then unit 2, ..., unit n, ..., designate [MAP(j) + $w(B_j)$ + 1 − k] as the address in array A of the k^{th} term of the form λ_{ij} where $d_{ij}^{+} = 1$; $k = 1, 2, \ldots, w(B_j)$.

The "MAP" array is generated recursively:

$$\text{MAP}(j+1) = \text{MAP}(j) + w(B_j) + 1,$$

with the initial condition that MAP(1) = 1.

Virtually identical techniques could be employed to store the matrix Λ by row rather than by column, if desired. For instance, one notes that row i of the matrix contains exactly $w(B_i)$ terms corresponding to *downward* transition rates (which are all unity), $[N - w(B_i)]$ terms that are upward transition rates, and an on-diagonal term λ_{ii}.

The coefficients stored in the collapsed matrix must be recovered for use in the equations of detailed balance (see Section 4). This is easily done since the state j is fixed for any particular equation; if $S_j = \{b_N, b_{N-1}, \ldots, b_1\}$, one simply starts with b_1 and the state $S_i^1 \equiv \{b_N, b_{N-1}, \ldots, b_1'\}$ where $b_1' \neq b_1$. Then λ_{ij} is unity if $b_1 = 0$ or $\lambda_{ij} = \text{A}[\text{MAP}(j) + w(B_j)]$ if $b_1 = 1$. One continues in this manner with $b_2, b_3, \ldots, b_l, \ldots, b_N$ and the associated adjacent vertices $S_i^2, S_i^3, \ldots, S_i^l, \ldots, S_i^N$. In general $S_i^l = \{b_N, b_{N-1}, \ldots, b_l', \ldots, b_1\}$ where $b_l' \neq b_l$, and $\lambda_{ij} = 1$ if $b_l = 0$ or $\lambda_{ij} = \text{A}[\text{MAP}(j) + w(B_j) + 1 - k]$ if $b_l = 1$ (k as defined in (2) above).

10. REDUCTION TO THE $M/M/N$ ZERO-LINE CAPACITY SYSTEM

The large size of the problem at hand is due to our demand to identify the busy and idle servers in the system's state description. However, it is clear that if the system is in some state B_i with $w(B_i)$ servers busy, the total transition rate "downward" to states with one less busy server is $w(B_i)$ and the total transition rate "upward" to states with one more busy server is λ [except for $w(B_i) = N$, in which case the upward transition rate is zero]. Thus, if we aggregate states according to the numbers of servers busy, we obtain the well-known $M/M/N$ zero-line capacity queuing system. For this system the steady-state probabilities are given as follows:

$$\text{prob}\{\text{exactly } n \text{ servers busy}\} = \frac{\lambda^n/n!}{\sum_{j=0}^{N} \lambda^j/j!}, \quad n = 0, 1, \ldots, N.$$

In terms of states of the hypercube, the event "exactly n servers busy" corresponds to the set of states on the n^{th} hyperplane

$$H_n = \{B \in C_N : w(B) = n\}, \quad n = 0, 1, 2, \ldots, N.$$

Thus, referring back to the equations of detailed balance, we can state the further condition that

$$P\{H_n\} = (\lambda^n/n!) \Big/ \sum_{j=0}^{N} \lambda^j/j!, \quad n = 0, 1, \ldots, N, \tag{7}$$

implying that the sum of probabilities on each hyperplane is known.

11. ITERATING TOWARD A SOLUTION

We are now ready to derive an iterative procedure for obtaining the steady state probabilities on the hypercube. Suppose $\hat{P}_n\{B_j\}$ is the estimate of $P\{B_j\}$ at the n^{th} iteration ($n = 0, 1, 2, \ldots$). Then, in a manner similar to point Jacobi iteration [Ref. 18, p. 57] we

use the equations of detailed balance to determine the values at successive iterations:

$$\hat{P}_n\{B_j\} = \left[\sum_{\{B_i \in C_N : d_{ij}^{+} = 1\}} \hat{P}_{n-1}\{B_i\}\lambda_{ij} + \sum_{\{B_i \in C_N : d_{\bar{ij}} = 1\}} \hat{P}_{n-1}\{B_i\} \right] \Big/ [\lambda + w(B_j)],$$
$$j = 1, 2, \ldots, 2^N - 2;\ n = 1, 2, \ldots \qquad (8)$$

(The probabilities $P\{B_0\}$ and $P\{B_{2^N-1}\}$ are known exactly from the $M/M/N$ model.)

In developing the details of the iteration procedure we are confronted with the problems of (1) providing good initial estimates of the steady state probabilities and (2) guaranteeing that the probabilities sum to unity [thereby satisfying equation (2)]. One can attack both problems simultaneously by exploiting the fact that the sum of probabilities along each hyperplane H_n is known from the $M/M/N$ birth and death model. We show that if one starts with estimated probabilities at iteration step 0 that sum to the exact birth and death values, then without any additional "error correcting" the sum of estimated probabilities along each hyperplane always equals the exact birth and death value, regardless of how deep one is into the iteration procedure. (Neglecting computer roundoff errors.) This initialization guarantees that equation (2) is satisfied without explicitly including equation (2) in the iteration process. It also has been found to yield probability estimates that converge rapidly to the correct values (within the tolerance specified by the convergence criterion).

We prove by induction that the sum of estimated probabilities along each hyperplane maintains its correct value. (It is assumed that at step 0 the procedure is set up with this property.) Let

$$P(k) = (\lambda^k/k!) \Big/ \sum_{j=0}^{n} \lambda^j/j!, \quad k = 0, 1, \ldots, N. \qquad (*)$$

Suppose for some iteration $n-1$ ($n > 0$) that the hyperplane probabilities sum to the correct values, i.e.,

$$\sum_{\{B_j \in H_k\}} \hat{P}_{n-1}\{B_j\} = P(k), \quad k = 0, 1, \ldots, N.$$

Then for iteration n,

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = P(k), \quad k = 0, 1, \ldots, N. \qquad (**)$$

Proof: (a) For $k = 0$ and $k = N$ the result is obvious, since $P\{B_0\}$ and $P\{B_{2^N-1}\}$ are maintained exactly at $P(0)$ and $P(N)$, respectively, for all n.

(b) For $k = 1, 2, \ldots, N-1$, we apply the iteration equation (8), obtaining

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k} \left\{ \sum_{\{B_j \in H_k\}} \left[\sum_{\{B_i \in C_N : d_{ij}^{+} = 1\}} \hat{P}_{n-1}\{B_i\}\lambda_{ij} + \sum_{\{B_i \in C_N : d_{\bar{ij}} = 1\}} \hat{P}_{n-1}\{B_i\} \right] \right\}.$$

Interchanging order of summations between adjacent hyperplanes, we have

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k} \left[\sum_{\{B_i \in H_{k-1}\}} \hat{P}_{n-1}\{B_i\} \sum_{\{B_j \in C_N : d_{ij}^{+} = 1\}} \lambda_{ij} + \right.$$
$$\left. + \sum_{\{B_i \in H_{k+1}\}} \hat{P}_{n-1}\{B_i\} \sum_{\{B_j \in C_N : d_{\bar{ij}} = 1\}} (1) \right].$$

But since the total upward transition rate from each state equals λ, we have for each vertex $B_i \in H_{k-1}$,

$$\sum_{\{B_j \in C_N : d_{ij}^{\sim} = 1\}} \lambda_{ij} = \lambda.$$

Also, since the total downward transition rate from each state in hyperplane l equals l, we have for each vertex $B_i \in H_{k+1}$,

$$\sum_{\{B_j \in C_N : d_{ij}^{\sim} = 1\}} 1 = k + 1.$$

Thus

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k}\left[\lambda \sum_{\{B_i \in H_{k-1}\}} \hat{P}_{n-1}\{B_i\} + (k+1) \sum_{\{B_i \in H_{k+1}\}} \hat{P}_{n-1}\{B_k\}\right].$$

Now, applying the inductive hypothesis, we can write

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k}[\lambda P(k-1) + (k+1)P(k+1)].$$

Since this last equation is contained in the set of "birth and death" equations whose solution is (*), we must have

$$\sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = P(k).$$

Examining the proof, we note that iterations could be performed first for even-numbered hyperplanes, then odd-numbered, then even-numbered, etc. Such a procedure of interleaving even- and odd-numbered hyperplanes provides each computation of (8) with the most recent available estimates of the unknowns (which for each equation are always in adjacent hyperplanes), thereby yielding a Gauss–Seidel procedure, which is known to be asymptotically faster than point Jacobi iteration.*

To start the iterative procedure we need to provide initial estimates for all probabilities in odd-numbered hyperplanes, i.e., we need $P_0\{B_j\}$ for all $\{B_j \in H_n : n = 1, 3, 5, \ldots, \}$. For simplicity we assume the probabilities of all states in any given hyperplane are *equal* and set

$$\hat{P}_0\{B_j\} = \left[\lambda^{w(j)} \Big/ \left([w(j)!] \cdot \sum_{i=0}^{N} \frac{\lambda^i}{i!}\right)\right] \Big/ \binom{N}{w(j)}, \tag{9}$$

recognizing the numerator as the appropriate steady-state probability of the corresponding $M/M/N$ system and the denominator as the binomial coefficient, the total number of states in $H_{w(j)}$. Then the iterations indicated by (8) are performed in an oscillating manner, first for even-numbered hyperplanes, then odd-numbered, etc. The iterations are terminated as soon as some convergence criterion is satisfied.

Round-off error is no problem with the proposed method. As is common with most matrix iterative techniques, round-off errors do not accumulate because one is using the same (exact) matrix of coefficients at each iteration. Thus, any round-off error is due only to the current iteration. More significant would be estimation errors whose values are determined by the convergence criterion.

* It is readily shown that the matrix Λ is *irreducibly diagonally dominant* [Ref. 18, p. 23] and therefore that both point Jacobi and point Gauss–Seidel iteration methods are convergent [Ref. 18, p. 73]. Moreover, the matrix Λ is readily shown to have a nonnegative *point Jacobi matrix* [Ref. 18, p. 57]. In such a case, a theorem of Stern and Rosenberg[19] states that point Gauss–Seidel iteration is asymptotically faster than point Jacobi iteration.

12. COMPUTED MEASURES OF PERFORMANCE: ZERO LINE CAPACITY

Once the numerical values for the state probabilities are obtained to a desired accuracy, it is necessary to compute the system's performance characteristics. The present model computes the following:

(1) Region-wide mean travel time.
(2) Region-wide workload imbalance.
(3) Region-wide fraction of dispatches which are interdistrict.
(4) Workload of each response unit.
(5) Mean travel time to each geographical atom.
(6) Mean travel time to each district.
(7) Mean travel time of each response unit.
(8) Fraction of responses into each district that are interdistrict.
(9) Fraction of responses of each response unit that are interdistrict.
(10) Fraction of all responses that result in unit n being dispatched to atom j.

Such a mixture of performance measures allows one to focus simultaneously on several region-wide objectives while assuring that spatial inequities in the delivery of service are maintained at an acceptable minimum.

We now develop the mathematical procedures for computing these performance measures for the zero line capacity system.

Individual workloads

The workload of unit n, designated ρ_n, is simply equal to the fraction of time that unit n is busy servicing calls. Thus ρ_n is equal to the sum of the steady-state probabilities on the hyperplane corresponding to $b_n = 1$, i.e.,

$$\rho_n = \sum_{\{i:b_n=1\}} P\{B_i\}.$$

The unit-step tour derived in Section 5 has certain periodicity properties that make the computation of the sum particularly easy.

Region-wide workload imbalance

There are several measures of workload imbalance:

Absolute deviation: $\Delta W = \underset{n,m}{\text{MAX}}|\rho_n - \rho_m|$.

Variance: $\sigma_W^2 = \sum_{n=1}^{N} \rho_n^2 - \left[\sum_{n=1}^{N} \rho_n\right]^2$.

Standard deviation: $\sigma_W = \sqrt{\sigma_W^2}$.

Percentage positive deviation from mean: $[\Delta W/\overline{W}]^+ = \left[\underset{n}{\text{MAX}}\left\{\frac{\rho_n}{\frac{1}{N}\sum_{j=1}^{N}\rho_j}\right\}\cdot 100\%\right] - 100\%$.

Percentage negative deviation from mean: $[\Delta W/\overline{W}]^- = 100\% - \left[\underset{n}{\text{MIN}}\left\{\frac{\rho_n}{\frac{1}{N}\sum_{j=1}^{N}\rho_j}\right\}\cdot 100\%\right]$.

For the remainder of the system performance characteristics, it is necessary to compute

$$\rho_{nj} = \text{fraction of all dispatches that send unit } n \text{ to geographical atom } j \quad \left(\sum_{n,j} \rho_{nj} = 1\right).$$

Using standard arguments* the formula is

$$\rho_{nj} = \frac{\sum_{\{B_i \in E_{nj}\}} \lambda f_j P\{B_i\}/\eta_{ij}}{\lambda(1 - P\{B_{2^N-1}\})}, \tag{11}$$

where E_{nj} is the set of states in which unit n is an optimal unit to assign to a call from atom j. The common factor λ in the numerator and the denominator can be ignored. We compute ρ_{nj} by retracing the unit-step hypercube tour, once for each atom. If, during the course of the tour associated with atom j, we are at some vertex B_i and have determined the total number of optimal units η_{ij} and their identities, $r_1, r_2, \ldots, r_{\eta_{ij}}$, then exactly η_{ij} of the ρ_{nj}'s are incremented:

$$\rho_{nj} \leftarrow \rho_{nj} + f_j P\{B_i\}/\eta_{ij}, \quad n = r_1, r_2, \ldots, r_{\eta_{ij}}, \tag{12}$$

neglecting (for the moment) the denominator of (11). After computations are completed for all (n, j), we scale the result:

$$\rho_{nj} \leftarrow \rho_{nj}/[1 - P\{B_{2^N-1}\}]. \tag{13}$$

Now we can state without further comment the formulas for the remaining performance measures.

Fraction of total dispatches that are interdistrict

$$FT = \sum_{n=1}^{N} \sum_{j \notin \text{district } n} \rho_{nj} \tag{14}$$

Fraction of dispatches to unit n that are interdistrict

$$FI_n = \frac{\sum_{j \notin \text{district } n} \rho_{nj}}{\sum_{j=1}^{J} \rho_{nj}} \tag{15}$$

Fraction of district i calls that require an out-of-district unit

$$FO_i = \frac{\sum_{n \neq i} \sum_{j \in \text{district } i} \rho_{nj}}{\sum_{n=1}^{N} \sum_{j \in \text{district } i} \rho_{nj}} \tag{16}$$

Unconditional mean travel time

$$\bar{T} = \sum_{n=1}^{N} \sum_{j=1}^{J} \rho_{nj} t_{nj} \tag{17}$$

* See, for example, Ref. [20].

Average travel time to atom j

$$\overline{T}_j = \frac{\sum_{n=1}^{N} \rho_{nj} t_{nj}}{\sum_{n=1}^{N} \rho_{nj}} \tag{18}$$

Average travel time to district i

$$\overline{TD}_i = \frac{\sum_{j \in \text{district } i} \sum_{n=1}^{N} \rho_{nj} t_{nj}}{\sum_{j \in \text{district } i} \sum_{n=1}^{N} \rho_{nj}} \tag{19}$$

Average travel time of unit n

$$\overline{Tu}_n = \frac{\sum_{j=1}^{J} \rho_{nj} t_{nj}}{\sum_{j=1}^{J} \rho_{nj}} \tag{20}$$

13. MODIFICATIONS FOR AN INFINITE-LINE CAPACITY SYSTEM

Most of the performance measures of the previous section can be easily calculated for an infinite-line capacity system, allowing for the appropriate modifications for the calls which incur a queue delay. It is clear that the equations of detailed balance [equation (1)] still hold on the hypercube, but the total probability [equation (2)] must include the "infinite tail" as well:

$$\sum_{i=0}^{2^N-1} P\{B_i\} + \sum_{j=1}^{\infty} P_Q\{j\} = 1, \tag{21}$$

where $P_Q\{j\} \equiv \text{prob}\{\text{exactly } j \text{ calls in queue, assuming steady state conditions}\}$.

The reduced birth and death process now contains a countably infinite number of states with the downward transition rate from all states involving a queue equal to N (since all N response units are simultaneously busy). The hyperplane probabilities required for initialization at the first iteration are accordingly reduced:

$$P\{H_n\} = (\lambda^n/n!) \Big/ \left[\sum_{j=0}^{N} \frac{\lambda^j}{j!} + \frac{\lambda^N}{N!} \frac{\lambda/N}{1 - \lambda/N} \right]. \tag{22}$$

The probabilities corresponding to a queue existing are given by

$$P_Q\{j\} = \frac{\dfrac{\lambda^N}{N!}\left(\dfrac{\lambda}{N}\right)^j}{\displaystyle\sum_{k=0}^{N} \frac{\lambda^k}{k!} + \frac{\lambda^N}{N!} \frac{\lambda/N}{1 - \lambda/N}}, \quad j = 1, 2, \ldots . \tag{23}$$

Since equation (1) still holds on the hypercube, the iterative solution technique developed in Sections 5–11 still applies, with the only modifications reflected by equations (21) and (22).

Other more significant modifications are required when computing the performance measures, which are addressed below in the same order as in Section 12.

Individual workloads

Let

$P_Q \equiv$ prob{a queue of length one or greater exists, assuming steady state conditions}.

Clearly,

$$P_Q = \sum_{i=1}^{\infty} P_Q\{i\}.$$

Now, since each unit works all of the time that a queue exists, an additional term must be added to equation (10) to compute the workload of unit n:

$$\rho_n = \sum_{\{j: b_n = 1\}} P\{B_j\} + P_Q. \tag{24}$$

Region-wide workload imbalance

All of the previous equations for workload imbalance still apply without modification, where ρ_n is now given by equation (24).

Fraction of dispatches that send a unit to a particular geographical atom

For the case of a queue we can split ρ_{nj} into two components,

$$\rho_{nj} = \rho_{nj}^{[1]} + \rho_{nj}^{[2]} \tag{25}$$

where $\rho_{nj}^{[1]}$ = fraction of all dispatches that send unit n to atom j *and* incur no queue delay,
$\rho_{nj}^{[2]}$ = fraction of all dispatches that send unit n to atom j and do incur a positive queue delay.

The first term is given by a modification of equation (11):

$$\rho_{nj}^{[1]} = \sum_{\{B_i \in E_{nj}\}} f_j P\{B_i\}/\eta_{ij}. \tag{26}$$

The term $\rho_{nj}^{[2]}$ is equal to the product of three terms: (1) the probability that a randomly arriving call incurs a queue delay; (2) the conditional probability that the call originated from atom j, given it incurs a queue delay; and (3) the conditional probability that the call results in the dispatch of unit n, given it originates from atom j and incurs a queue delay. Clearly a queue delay will be incurred by any call arriving while all N response units are busy, and thus the first term is

$$P'_Q \equiv P_Q + P\{B_{2^N-1}\}.$$

The second term is equal to the fraction of calls f_j that are generated from atom j, and is not dependent on the fact that a queue exists. To obtain the third term we use the fact that queued calls are handled FCFS, regardless of the call's location or the responding unit's location (at the scene of a previous call); thus, any of the N busy units is equally likely to be assigned to any particular queued call, yielding a conditional probability of $1/N$. Summarizing, we have

$$\rho_{nj}^{[2]} = f_j P'_Q/N. \tag{27}$$

Interdistrict dispatches

Given the new definition for ρ_{nj}, reflected in equations (25), (26) and (27), equations (14), (15) and (16) regarding interdistrict dispatches apply without modification.

Unconditional mean travel time

To compute the unconditional mean travel time we define the "queued call travel time",

$$\overline{T}_Q \equiv \sum_{i=1}^{J} \sum_{j=1}^{J} f_i f_j \tau_{ij}. \tag{28}$$

To interpret this expression, consider any particular call which incurs a queue delay. The call is generated from atom j with probability f_j. Each one of the N busy response units is equally likely to be dispatched to the call. The probability that the dispatched unit must travel from atom i is f_i, the fraction of region-wide workload generated from atom i, and the dispatch assignment is made independently of the particular values assumed by i and j. Thus, with conditional probability $f_i f_j$, the travel time to a queued call will be the travel time from atom i to atom j, τ_{ij}. Hence equation (28) is the mean travel time to a call that incurs a queue.

The expression analogous to equation (17), giving the region-wide unconditional mean travel time, can now be written:

$$\overline{T} = \sum_{n=1}^{N} \sum_{j=1}^{J} \rho_{nj}^{[1]} t_{nj} + P_Q' \overline{T}_Q. \tag{29}$$

Average travel time to atom j

Following reasoning analogous to that above, we write

$$\overline{T}_j = \frac{\sum_{n=1}^{N} \rho_{nj}^{[1]} t_{nj}}{\sum_{n=1}^{N} \rho_{nj}^{[1]}} (1 - P_Q') + \sum_{i=1}^{J} f_i \tau_{ij} P_Q'. \tag{30}$$

Average travel time to district i

Again, following the above reasoning,

$$\overline{TD}_i = \frac{\sum_{j \in \text{district } i} \sum_{n=1}^{N} \rho_{nj}^{[1]} t_{nj}}{\sum_{j \in \text{district } i} \sum_{n=1}^{N} \rho_{nj}^{[1]}} (1 - P_Q') + \frac{\sum_{k \in \text{district } i} \sum_{j=1}^{J} f_j f_k \tau_{jk}}{\sum_{k \in \text{district } i} f_k} P_Q'. \tag{31}$$

Average travel time of unit n

Unfortunately the author knows of no exact analytical expression for the infinite line-capacity system, analogous to equation (20), for the average travel time of unit n. The problem in deriving such as expression arises from the fact that a unit's position when dispatched for the first time back-to-back during a system saturation period (i.e., a period of positive queue length) is not selected from the probability distribution of call locations f_j. Such a unit was most probably assigned to its current call when *several* units were available, and

thus its location tends to be near its home location (or district). As an approximation we compute the mean travel time for unit n as follows:

$$\overline{Tu}_n = \frac{\sum_{j=1}^{J} \rho_{nj}^{[1]} t_{nj} + T_Q P'_Q/N}{\sum_{j=1}^{J} \rho_{nj}^{[1]} + P'_Q/N}. \tag{32}$$

We recognize that the term reflecting travel time to queued calls is an overestimate which becomes asymptotically exact as the system utilization factor $\rho = \lambda/N \to 1$.

14. ILLUSTRATIVE COMPUTATIONAL RESULTS

To date we have limited computational experience using a PL/I program that implements the techniques of the previous sections. The program is written with fully variable storage that adjusts array sizes to match the requirements implied by N and J. Further details of the computer program are found in a forthcoming document.

As an illustration of the use of the program, consider a nine-district linear command with 18 equal-sized atoms, using an $\overline{\text{MCM}}$ dispatching strategy (see Fig. 3). It is assumed that calls are distributed uniformly over the command and that response units are dispatched from mobile positions, with each unit's position (while idle) selected from a uniform distribution over the unit's district, which has unit length. Comparable runs have been made with a zero-line capacity system and an infinite-line capacity system. We found that each run costs less than \$3.00 on the M.I.T. IBM 370 model 155. In these runs the convergence criterion was a rather simple one: no state probability shall change by more than 10^{-5} between successive iterations; as soon as this criterion is satisfied, the iterations are stopped and summary statistics are calculated. To achieve this convergence objective the algorithm never required more than 11 iterations. The average number of iterations was approximately 8. Of course, the required number of iterations would increase if the convergence criterion were made more stringent.

Using the results of these runs, the average command-wide average travel distance versus $\rho(\equiv\lambda/9)$ is plotted in Fig. 4 for both the zero-line capacity and the infinite-line capacity systems. One sees that this travel distance starts at $E[D] = 1/3$ at $\rho = 0$ (as expected from elementary considerations [Ref. 4, Chap. 3]) and increases monotonically with ρ. The average travel distance of the infinite-line capacity system clearly becomes greater than that of the zero-line capacity starting at $\rho \approx 0{\cdot}4$. The mean travel distance of the latter system remains relatively smaller because of calls which are lost when the system is saturated; in the infinite-line capacity system calls that arrive when the system is saturated enter a queue and require a mean travel distance of 3.

The fraction of dispatches to each unit which are interdistrict is shown as a function of ρ in Fig. 5a and 5b. The "end effects" of system behavior are particularly apparent in these

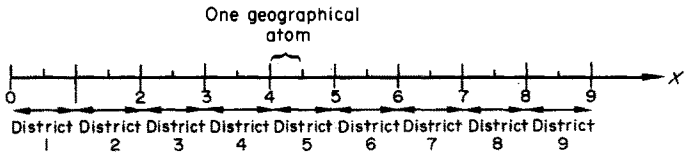

Fig. 3. Nine-district homogeneous linear command; $\overline{\text{MCM}}$ dispatching; mobile units.

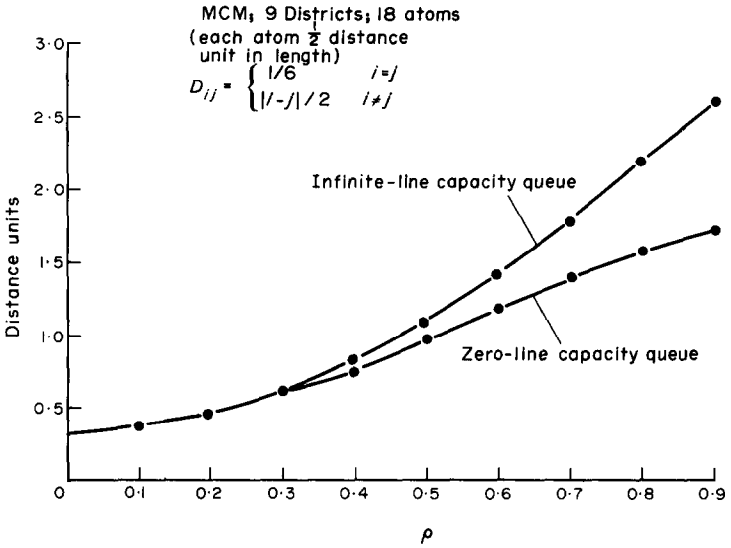

Fig. 4. Average command-wide travel distance ($D_{ij} \equiv$ travel distance between atoms i and j, $i, j = 1, 2, \ldots, 18$).

figures. For instance, at $\rho = 0{\cdot}1$ units 1 and 9 are sent out of their respective districts for only about 5 per cent of their dispatch assignments, whereas the adjacent units 2 and 8, respectively, experience a fraction of interdistrict dispatches more than twice as great. It is interesting that the "curves" in Fig. 5a and 5b, which are symmetric about unit 5, are bimodal

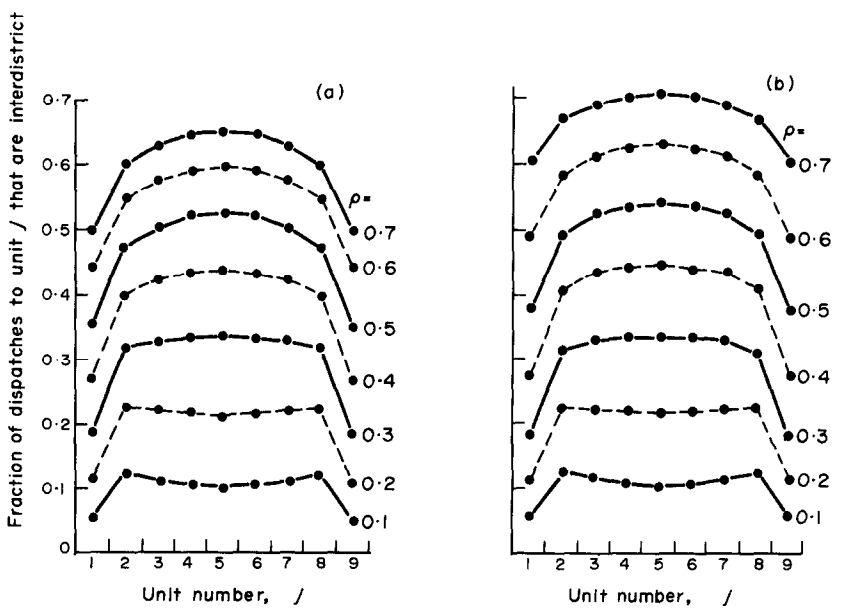

Fig. 5. Interdistrict dispatches—nine units, 18 atoms, $\overline{\text{MCM}}$ (a) Zero-line capacity queue, (b) Infinite-line capacity queue.

for small ρ but become unimodal for moderate and large values of ρ. Thus, as ρ gets large the center unit (unit 5) experiences the greatest fraction of interdistrict dispatches.

Figures 6a and 6b depict workloads of each unit as a function of ρ. One sees that these figures have similar properties to those of Fig. 5a and 5b.

We also have limited computational experience with $N = 10$, 11 and 12, and the cost per run has yet to exceed \$15. Thus the anticipated costs appear to be very reasonable considering the volume of useful information obtained.

15. EXTENSIONS AND GENERALIZATIONS

Now that it appears computationally feasible to examine spatially distributed queuing systems with up to 12 response units, it is natural to inquire into certain optimization problems of these systems. One such problem is the generalized district design problem, which in the sense of the hypercube model is a decision rule for which unit to dispatch to a call from atom j, given a particular system state; such a rule must be determined for each state and each atom j. Thus, a "districting" for the problem at hand is actually a set of up to $2^N - 1$ partitionings of the region, one for each nonsaturated system state. The model reported in this paper generates such partitionings, but only for the class of "fixed preferences" dispatch policies. Carter, Chaiken and Ignall[3] determined more general rules analytically for the case $N = 2$, so as to minimize time-average system mean travel time, but larger N appeared analytically intractable. Thus, one direction in which we hope to develop the model is toward algorithmic techniques for generating globally optimal dispatch decision rules, or districtings.

The districting problem assumes a set of initial locations, at least for the case of units assigned to fixed facilities. One would also want to develop the model in the direction of improving the initial locations of units. We also hope to do this, most probably using algorithmic approaches.

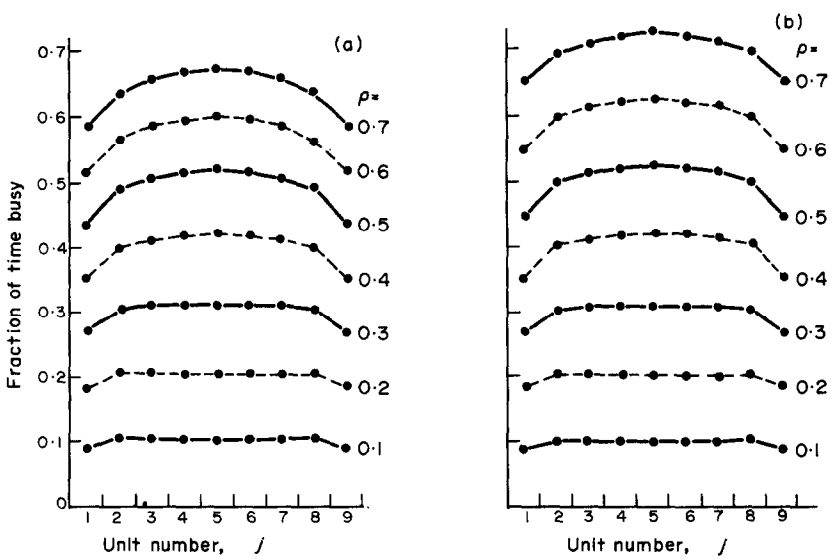

Fig. 6. Workload of units—nine districts, 18 atoms, $\overline{MCM}$ (a) Zero-line capacity queue, (b) Infinite-line capacity queue.

There are more fundamental extensions and generalizations of the model—and these relate to the basic model assumptions. For instance, the assumption of identical exponentially distributed service times might be troublesome in some applications, say where one unit consistently takes longer to service calls than others or where travel time is a nonnegligible fraction of the total service time. It is straightforward to replace each of the constant service rates (μ) with service rates that are state-dependent; this only "costs" a doubling of the computer storage required for the state transition matrix. With such a capability, one could, for instance, give each response unit n its "own" mean service time μ_n^{-1}.

Another problem involves the lack of treatment of call priorities. However, within the hypercube framework it is possible to allow a limited amount of attention to priorities. For instance, one could define f_{jk} to be the fraction of all calls that are generated from atom j *and* are priority k. Then, the dispatch policy could depend on priority level. However, the dispatch policy cannot be of a form that allows calls to be queued while units are still available (otherwise the special structure of the hypercube state space is destroyed). Similarly, one could have queued calls separated by priority and handled FCFS by priority class; but again, analogous to the case of the simple $M/M/N$ infinite-line capacity queue, order of service cannot depend on the identity of the server or the location of the call.

An application of the model to police operations in the greater Boston area is given in Ref. [21].

Acknowledgements—This work was supported in part by the U.S. Department of Housing and Urban Development under a contract to the New York City Rand Institute and in part by the National Science Foundation (Division of Social Systems and Human Resources) under a grant to the M.I.T. Operations Research Center. The author thanks Jan Chaiken and Saul Gass for suggesting changes in an earlier draft.

REFERENCES

1. J. M. Chaiken and R. C. Larson, Methods for allocating urban emergency units: A survey, *Management Sci.* **19**, 110–130 (1972).
2. C. Revelle, D. Marks and J. C. Liebman, An analysis of private and public sector location models, *Management Sci.* **11**, 692–707 (1970).
3. R. D. Smith, Computer applications in police manpower distribution, Field Service Division, International Association of Chiefs of Police, Washington, D.C. (1961).
4. R. C. Larson, *Urban Police Patrol Analysis*, The MIT Press, Cambridge, Mass. (1972).
5. S. Gass, On the division of police districts into patrol beats, In *Proceedings of the 23rd National Conference of the Association for Computing Machinery*, Brandon/Systems Press, Princeton (1968).
6. Hess *et al.*, Nonpartisan political redistribution by computer, *Ops Res.* **13**, 462–475 (1965).
7. W. Barnett and J. R. DuBois, The use of probability theory in the assignment of police patrol areas, National Institute of Law Enforcement and Criminal Justice, Law Enforcement Assistance Administration, U.S. Department of Justice PR 70-2 (1970).
8. L. C. Santone and G. N. Berlin, A computer model for the evaluation of fire station location, National Bureau of Standards Report, U.S. Department of Commerce, Washington, D.C. (1969).
9. J. B. Schneider and J. G. Symons, Locating ambulance dispatch centers in an urban region: A man-computer interactive problem-solving approach, RSRI Discussion Paper Series 49, Regional Science Research Institute, Philadelphia (1971).
10. J. B. Schneider, Solving urban location problems: Human intervention versus the computer, *J. Am. Inst. Planners* **37**, 95–99 (1971).
11. C. Toregas, R. Swain, C. Revelle and L. Bergman, The location of emergency service facilities, *Ops Res.* **19**, 1363–1373 (1971).
12. S. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph, *Ops Res.* **12**, 450–459 (1964).
13. S. Hakimi, Optimum distribution of switching centers in communications networks and some related graph theoretic problems, *Ops Res.* **13**, 462–475 (1965).
14. G. M. Carter, J. M. Chaiken and E. Ignall, Response areas for two emergency units, *Ops Res.* **20**, 571–594 (1972).

15. R. C. Larson and K. A. Stevenson, On insensitivities in urban redistricting and facility location, *Ops Res.* **20**, 595–612 (1972).
16. G. Campbell, A spatially distributed queuing model with application to police sector design, Master's Thesis in Operations Research, MIT (1972).
17. P. M. Morse, *Queues, Inventories and Maintenance*, The MIT Press, Cambridge, Mass. (1956).
18. R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, New Jersey (1962).
19. P. Stern and R. L. Rosenberg, On the solution of linear simultaneous equations by iteration, *J. Lond. Math. Soc.* **23**, 111–118 (1948).
20. R. E. Strauch, When a queue looks the same to an arriving customer as to an observer, *Management Sci.: Theory* **17**, 140 (1970).
21. R. C. Larson, Illustrative police sector redesign in district 4 in Boston, to appear in *Urban Analysis* **2** (1) (1974).

(*Paper received* 7 *March* 1973)